

002890034AA

31

CLAIMS

Having thus described our invention, what we claim as new and desire to secure by Letters Patent is as follows:

- 1 1. A method for generating database application conversion
- 2 automatically for tables in a database, the method comprising the following
- 3 steps:
- 4 invoking a computer application program to read table definitions
- 5 from the database;
- 6 the computer application program using existing database definitions
- 7 to define object classes and deployment descriptors; and
- 8 the computer application program producing a client-side helper class
- 9 to coordinate the name of a database used in a Naming and Directory Service
- 10 with client code.
- 1 2. The method of claim 1 further comprising the following steps:

FS-00534

002890034AA

32

2 utilizing a user interface to permit the user to interact with the
3 computer application program;
4 collecting data;
5 generating files in a user-specified directory path;
6 generating a Data JAVA file, a Home Interface JAVA file, a Remote
7 Interface JAVA file, a Bean JAVA file, a Primary Key JAVA file, a Persistent
8 JAVA file, an Enterprise JAVA Bean Deployment Descriptor XML file, an
9 Enterprise JAVA Bean Jar batch command file, a Vendor-Specific
10 Deployment XML file, and a Vendor-Specific Build batch command file.

1 3. The method of claim 2 for generating the data collection file comprising
2 the following steps:
3 the computer application program querying the database to get names
4 of all tables related to the database for the user to encapsulate the tables with
5 the EJB; and
6 querying the database to acquire information about fields within each
7 table the user has selected.

1 4. The method of claim 2 for generating the Data JAVA file comprising the
2 following steps:

FS-00534

002890034AA

002890034AA

33

3 generating a file with its filename equal to a table name followed by
4 the word "Data" and file extension ".java";
5 writing header comments indicating that a JAVA class encapsulates
6 one row of the selected table;
7 writing a JAVA package definition consistent with the user-specified
8 directory path;
9 writing a first JAVA statement to import a java.io.Serializable class;
10 writing a JAVA class definition with a class name equal to its
11 filename;
12 writing a second JAVA statement indicating that the class implements
13 the Serializable interface;
14 writing a corresponding attribute definition for each one of a plurality
15 of fields in the selected table;
16 ensuring that a JAVA data type of each one of a plurality of attributes
17 is compatible with a database data type of its corresponding field;
18 ensuring that a name of each one of the plurality of attributes is
19 identical to the name of its corresponding field;
20 writing a first JAVA public method to get a value of each one of the
21 plurality of attributes;

FS-00534

T002890034AA

002890034AA

34

22 writing a second JAVA public method to set the value of each one of
23 the plurality of attributes; and
24 writing a third JAVA public method named "toString" to return a
25 string representing a Data class.

1 5 The method of claim 2 for generating the Home Interface JAVA file
2 comprising the following steps;
3 for each one of the plurality of selected tables, generating a file with
4 filename equal to a table name followed by the word "Home" and file
5 extension ".java";
6 writing header comments indicating that a JAVA class is the Home
7 Interface for the Bean encapsulating the selected table,
8 writing a JAVA package definition consistent with the user-specified
9 directory path;
10 writing a first JAVA statement to import the following classes:
11 javax.ejb.EJBHome; javax.ejb.FinderException; javax.ejb.CreateException,
12 java.rmi.RemoteException; java.util.Collection,
13 writing a JAVA interface definition with an interface name equal to
14 the filename;

FS-00534

FOOSE/03/07/2001

002890034AA

35

15 writing a second JAVA statement indicating that the class extends the
16 EJBHome interface;
17 writing a first JAVA method signature named "create" which takes as
18 parameters the attributes of Data class and returns an object of Remote class;
19 writing a second JAVA method signature named "create" which takes
20 as a parameter an object of Data class and returns an object of Remote class;
21 and
22 writing a third JAVA method signature named "findByPrimaryKey"
23 which takes as a parameter an object of PrimaryKey class and returns an
24 object of Remote class wherein writing the JAVA method signature named
25 "findByPrimaryKey" has one primary key and the computer application
26 program makes the parameter its corresponding attribute of Data class.

1 6 The method of claim 2 for generating the Remote Interface JAVA file
2 comprising the following steps:
3 generating, for each selected table, a file with its filename equal to the
4 table name followed by the word "Remote" and file extension "java";
5 writing header comments indicating that this JAVA class is the
6 Remote interface for the Bean encapsulating the selected table;

FS-00534

091616-073001

002890034AA

36

7 writing a JAVA package definition consistent with the user-specified
8 directory path;
9 writing a first JAVA statement to import the following classes:
10 javax.ejb.EJBObject; java.rmi.RemoteException;
11 writing a JAVA interface definition with its interface name equal to
12 the filename;
13 writing a second JAVA statement indicating that the class extends the
14 EJBObject interface;
15 writing a first JAVA public method signature to get a value of each
16 attribute and wherein the signature indicates that the method may throw a
17 Remote Exception; and
18 writing a second JAVA public method to set the value of each attribute
19 signature and wherein the signature shall indicate that the method may throw
20 a Remote Exception.

1 7. The method of claim 2 generating the Bean JAVA file comprising the
2 following steps:
3 generating, for each selected table, a file with its filename equal to its
4 table name followed by the word "Bean" and file extension "java";

FS-00534

002890034AA

37

5 writing header comments indicating that the JAVA class is an EJB

6 which encapsulates one row of the selected table;

7 writing a JAVA package definition consistent with the user-specified

8 directory path;

9 writing a first JAVA statement to import the following classes:

10 java.rmi.RemoteException; all classes in the javax.ejb package; all classes in

11 the java.util package;

12 writing a JAVA class definition with its class name equal to the

13 filename;

14 writing a second JAVA statement indicating that the class implements

15 the EntityBean interface;

16 writing an attribute definition for one attribute named "theContext" of

17 type "EntityContext";

18 writing, for each field in the selected table, its corresponding attribute

19 definition;

20 ensuring that the JAVA data type of each attribute is compatible with

21 the database data type of its corresponding field;

22 ensuring that each attribute is identical to the lower-case name of its

23 corresponding field;

24 writing a first JAVA public method to get the value of each attribute;

FS-00534

2003-09-29 09:59:59

002890034AA

38

```
25         writing a second JAVA public method to set the value of each
26         attribute;
```

27 writing a third JAVA public method named "setAll" which has an
28 object of Data class as parameter and can throw a Remote Exception and
29 wherein each attribute of Bean class is set to the value of its corresponding
30 Data attribute;

```
31      writing a fourth JAVA public method named "getAll" which returns
32      an object of Data class and can throw a Remote Exception;
```

33 writing a fifth JAVA public method named "ejbCreate" with attributes
34 of Data class as parameters and a signature of this method says it returns an
35 object of type "String" but the body of the method must return "null" and this
36 method can throw a Create Exception and a Remote Exception;

37 writing a sixth JAVA public method named "ejbPostCreate" with
38 attributes of Data class as parameters and the signature of this method says it
39 returns an object of type "String" but the body of the method must return
40 "null" and this method can throw a Create Exception and a Remote Exception;

```
41         writing a seventh JAVA public method named "ejbPostCreate" with an
42         object of Data class as the parameter and the signature of this method says it
43         returns an object of type "String" but the body of the method must return
44         "null" and this method can throw a Create Exception and a Remote Exception;
```

FS-00534

002890034AA

39

45 writing an eighth JAVA public method named "ejbLoad" which can
46 throw an EJB Exception or a Remote Exception and the body of this method
47 may be empty;

48 writing a ninth JAVA public method named "ejbStore" which can
49 throw an EJB exception or a Remote exception and the body of this method
50 may be empty;

51 writing a tenth JAVA public method named "ejbActivate" which can
52 throw an EJB Exception or a Remote Exception and the body of this method
53 can be empty;

54 writing an eleventh JAVA public method named "ejbPassivate" which
55 can throw an EJB Exception or a Remote Exception and the body of this
56 method can be empty;

57 writing a twelfth JAVA public method named "ejbRemove" which can
58 throw an EJB Exception or a Remote Exception and the body of this method
59 can be empty;

60 writing a thirteenth JAVA public method named "setEntityContext"
61 which can throw an EJB exception or a Remote Exception and the parameter
62 for this method is an object of type EntityContext and is used to set
63 theContext;

FS-00534

TOP SECRET

002890034AA

40

```
64         writing a fourteenth JAVA public method named "unsetEntityContext"
65         which can throw an EJB Exception or a Remote Exception and this method
66         sets theContext to "null", and
67         writing a fifteenth JAVA public method named "toString" to return a
68         string representing the Bean class.
```

1 8. The method of claim 2 for generating the Primary Key JAVA file
2 comprising the following steps:
3 generating a file for each selected table with at least two primary key
4 fields such that its filename is made equal to its table name followed by the
5 word "PrimaryKey" and file extension ".java";
6 writing header comments indicating that this JAVA class encapsulates
7 the Primary Key of the selected table;
8 writing a JAVA package definition consistent with the user-specified
9 directory path;
10 writing a first JAVA statement to import the java.io.Serializable class;
11 writing a JAVA class definition with the class name equal to its
12 filename;
13 writing a second JAVA statement indicating that its class implements
14 the Serializable interface;

FS-00534

002890034AA

41

15 writing a corresponding attribute definition for each primary key field;
16 ensuring that the JAVA data type of each attribute is compatible with
17 the database data type of its corresponding field;
18 ensuring that each attribute is identical to the lower case name of its
19 corresponding field;
20 writing a first JAVA public method to get the value of each attribute;
21 writing a second JAVA public method to set the value of each
22 attribute;
23 writing a third JAVA public method named "equals" which returns a
24 boolean value and the parameter to this method is an object of the Object class
25 and returns "true" if the parameter is an instance of the Primary Key class and
26 if the parameter's attributes have values equal to the values of the attributes of
27 this object; and
28 writing a fourth JAVA public method named "hashCode" which
29 returns an integer value and this method forms a String of the attribute values
30 and returns a hash code of that String.

1 9. The method of claim 2 for generating the Persistent JAVA file comprising
2 the following steps:

FS-00534

002890034AA

42

```

3         generating, for each selected table, a file with filename equal to the
4         table name followed by the word "Persistent" and file extension ".java";
5         writing header comments indicating that this JAVA class encapsulates
6         one row of the selected table and this client side class can be passed to the
7         server for execution;
8         writing a JAVA package definition consistent with the user specified
9         directory path;
10        writing a first JAVA statement to import the following classes
11        javax.naming.InitialContext; javax.naming.NamingException;
12        javax.rmi.PortableRemoteObject; java.sql.Connection; java.io.Serializable; all
13        classes of the utility package com.lmco.util;
14        writing a JAVA class definition with the class name equal to the
15        filename;
16        writing a second JAVA statement indicating that the class extends its
17        corresponding Data class;
18        writing a third JAVA statement indicating that the class implements
19        the Serializable interface and the PersistentObject interface;
20        writing a first JAVA public method to construct an object of this class
21        without any parameters;

```

FS-00534[illegible]

002890034AA

43

22 writing a second JAVA public method to construct an object of this
23 class with the attributes of the Data class as parameters;
24 writing a third JAVA public method to construct an object of this class
25 with an object of the Data class as parameters;
26 writing a fourth JAVA public method named "create" which returns a
27 boolean and the body of this method provides a JAVA try-and-catch block to
28 invoke the "create" method of the Home interface with the attributes of this
29 class as parameters and if an exception occurs, the result is false or otherwise
30 the result is true;
31 writing a fifth JAVA public method named "read" which returns a
32 boolean and the body of this method provides a JAVA try-and-catch block to
33 invoke the "findByPrimaryKey" method of the Home interface with the
34 Primary Key class as parameter and if an exception occurs, the result is false
35 or otherwise, the result is true and the attributes of this class are set to the
36 values in the Remote interface returned by the "findByPrimaryKey" method;
37 writing a sixth JAVA public method named "update" which returns a
38 boolean and the body of this method provides a JAVA try-and-catch block to
39 invoke the "findByPrimaryKey" method of the Home interface, with the
40 primary key class as parameter and if an exception occurs, the result is false

FS-00534

T00640" 9797660

002890034AA

44

41 or otherwise, the result is true and the attributes of the Remote interface are
42 set to the values of the attributes in this class;
43 writing an seventh JAVA public method named "delete" which returns
44 a boolean and the body of this method provides a JAVA try-and-catch block
45 to invoke the "remove" method of the Home interface with the Primary Key
46 class as parameter and if an exception occurs, the result is false or otherwise
47 the result is true; and
48 writing a JAVA protected method named "getHome" to return an
49 object of the Home class and the body of this method providing a JAVA try-
50 and-catch block to use an object of type InitialContext to look up the JNDI
51 name which JNDI name is equal to the name of the selected table.

1 10. The method of claim 2 for generating the EJB Deployment Descriptor
2 XML file comprising the following steps.

3 generating a file with filename "ejb-jar.xml" and located in a folder
4 named META-INF one directory deep within the user-specified target path;
5 writing an XML header statement for documents of type "ejb-jar";
6 writing a first tag <ejb> to begin the document;
7 writing a second tag <description> to begin a description;

FS-00534

F00E40" 9T89T660

002890034AA

45

8 writing the description of a jar file which includes the package name
9 and the jar filename specified by the user;
10 writing a third tag `</description>` to end the description;
11 writing a fourth tag `<enterprise-beans>` to begin a list of EJB's;
12 writing a fifth tag `<entity>` to begin a definition of its corresponding
13 entity EJB;
14 writing a sixth tag `<description>` to begin the description of the EJB;
15 writing a description of the EJB including the name of its
16 corresponding selected table;
17 writing a seventh tag `</description>` to end the description;
18 writing an eighth tag `<ejb-name>` to begin the name of the EJB;
19 writing the name of its corresponding selected table;
20 writing a ninth tag `</ejb-name>` to end the name of the EJB;
21 writing a tenth tag `<home>` to begin a home of the EJB;
22 writing a fully-qualified-name of its corresponding Home class;
23 writing an eleventh tag `</home>` to end the home of the EJB;
24 writing a twelfth tag `<remote>` to begin a remote interface of the EJB;
25 writing a fully-qualified-name of its corresponding Remote class;
26 writing a thirteenth tag `</remote>` to end the remote of the EJB;
27 writing a fourteenth tag `<ejb-class>` to begin a bean class of the EJB;

FS-00534

| Variable | Mean | SD | Min | Max |
|---------------------------------|-------------|------|-----|-----|
| Age | 35.2 | 12.5 | 18 | 65 |
| Gender | Male | 10.5 | 0 | 20 |
| Marital status | Married | 15.2 | 0 | 25 |
| Education | High school | 12.5 | 0 | 20 |
| Occupation | Manager | 10.5 | 0 | 20 |
| Income | High | 15.2 | 0 | 25 |
| Health status | Good | 10.5 | 0 | 20 |
| Stress level | Low | 10.5 | 0 | 20 |
| Life satisfaction | High | 15.2 | 0 | 25 |
| Work-life balance | Good | 10.5 | 0 | 20 |
| Family support | High | 15.2 | 0 | 25 |
| Community involvement | High | 15.2 | 0 | 25 |
| Volunteer work | High | 15.2 | 0 | 25 |
| Charitable contributions | High | 15.2 | 0 | 25 |
| Political participation | High | 15.2 | 0 | 25 |
| Religious participation | High | 15.2 | 0 | 25 |
| Cultural participation | High | 15.2 | 0 | 25 |
| Artistic participation | High | 15.2 | 0 | 25 |
| Sports participation | High | 15.2 | 0 | 25 |
| Gardening participation | High | 15.2 | 0 | 25 |
| Reading participation | High | 15.2 | 0 | 25 |
| Travel participation | High | 15.2 | 0 | 25 |
| Volunteer work (hours) | 10.5 | 10.5 | 0 | 20 |
| Charitable contributions (\$) | 15.2 | 15.2 | 0 | 25 |
| Political participation (times) | 15.2 | 15.2 | 0 | 25 |
| Religious participation (times) | 15.2 | 15.2 | 0 | 25 |
| Cultural participation (times) | 15.2 | 15.2 | 0 | 25 |
| Artistic participation (times) | 15.2 | 15.2 | 0 | 25 |
| Sports participation (times) | 15.2 | 15.2 | 0 | 25 |
| Gardening participation (times) | 15.2 | 15.2 | 0 | 25 |
| Reading participation (times) | 15.2 | 15.2 | 0 | 25 |
| Travel participation (times) | 15.2 | 15.2 | 0 | 25 |

002890034AA

46

28 writing the fully-qualified-name of its corresponding Bean class;
29 writing a fifteenth tag `</ejb-class>` to end the bean class of the EJB;
30 specifying container-maintained persistence with the statement
31 `<persistence-type>Container</persistence-type>`;
32 writing a sixteenth tag `<prim-key-class>` to begin the primary key
33 class of the EJB;
34 writing the fully-qualified-name of its corresponding PrimaryKey
35 class;
36 writing a seventeenth tag `</prim-key-class>` to end the primary key
37 class of the EJB;
38 specifying the EJB as not re-entrant by writing
39 `<reentrant>False</reentrant>`;
40 beginning each field of the selected table with `<cmp-field>``<field-`
41 `name>`;
42 writing each field of the selected table;
43 ending each field of the selected table with `</field name>``</cmp-`
44 `field>`;
45 if the selected table has only one primary key field, writing the
46 primary key field as `<primkey-field>` followed by the field name followed by
47 `</primkey-field>`;

FS-00534

T00E20" 9T9T660

002890034AA

47

48 writing an eighteenth tag </entity> to end a description of its
49 corresponding entity EJB for each selected table;
50 writing a nineteenth tag <assembly-descriptor> to begin the assembly
51 descriptor;
52 writing a twentieth or more tags defining a default security role;
53 providing default permission to all methods of all EJB's;
54 specifying container-managed transactions for all methods of each
55 EJB;
56 writing a twenty-first or greater tag <assembly-descriptor> to end
57 assembly;
58 writing a twenty-second or greater tag </enterprise-beans> to end the
59 list of EJB's; and
60 writing a twenty-third or greater tag </ejb> to end the document.

1 11. The method of claim 2 for generating the EJB Jar batch command file
2 comprising the following steps:
3 generating a file with filename set to the user-specified name of the Jar
4 file where the file extension on MS Windows-based systems is "bat";
5 writing a command to compile each generated Java file;
6 writing a command to put the ejb-jar.xml file into one Jar file; and

FS-00534

002890034AA

48

7 for each selected table, writing a command to put all of generated
8 JAVA classes into the Jar file.

1 12. The method of claim 2 for generating the Vendor-Specific Deployment
2 XML file comprising the following steps:

3 generating a file with filename indicating both the vendor and the
4 user-specified name of the EJB Jar file;
5 generating one or more XML tags according to vendor specifications
6 for a target Application Server and for which the details will vary with the
7 Vendor;

8 generating one or more XML tags to specify a JNDI name for the EJB
9 corresponding to each selected table and the JNDI name is the same as its
10 selected table name; and
11 generating one or more XML tags to specify mapping of each EJB
12 attribute to its field in the selected database table.

1 13. The method of claim 2 for generating the Vendor-Specific Build batch
2 command file comprising the following steps:

FS-00534

002890034AA

49

- 3 generating a file with its filename indicating both a vendor and the
- 4 word "command" and wherein the file extension on MS Windows-based
- 5 systems is "bat";
- 6 generating vendor-specific commands to import and deploy a Jar file
- 7 according to the Vendor-Specific Deployment file.

2007-07-25 09:51:50

FS-00534